ClickHouse 面试题

1、什么是 ClickHouse？

ClickHouse 是近年来备受关注的开源列式数据库管理系统，主要用于数据分析（OLAP）领域。通过向量化执行以及对 cpu 底层指令集（SIMD）的使用，它可以对海量数据进行并行处理，从而加快数据的处理速度。ClickHouse 从 OLAP 场景需求出发，定制开发了一套全新的高效列式存储引擎，并且实现了数据有序存储、主键索引、稀疏索引、数据 Sharding、数据 Partitioning、TTL、主备复制等丰富功能。

2、ClickHouse 有哪些应用场景？

1. 绝大多数请求都是用于读访问的；
2. 数据需要以大批次（大于 1000 行）进行更新，而不是单行更新；
3. 数据只是添加到数据库，没有必要修改；
4. 读取数据时，会从数据库中提取出大量的行，但只用到一小部分列；
5. 表很“宽”，即表中包含大量的列；
6. 查询频率相对较低（通常每台服务器每秒查询数百次或更少）；
7. 对于简单查询，允许大约 50 毫秒的延迟；
8. 列的值是比较小的数值和短字符串（例如，每个 URL 只有 60 个字节）；
9. 在处理单个查询时需要高吞吐量（每台服务器每秒高达数十亿行）；
10. 不需要事务；

11. 数据一致性要求较低；
12. 每次查询中只会查询一个大表。除了一个大表，其余都是小表；
13. 查询结果显著小于数据源。即数据有过滤或聚合。返回结果不超过单个服务器内存。

3、ClickHouse 列式存储的优点有哪些？

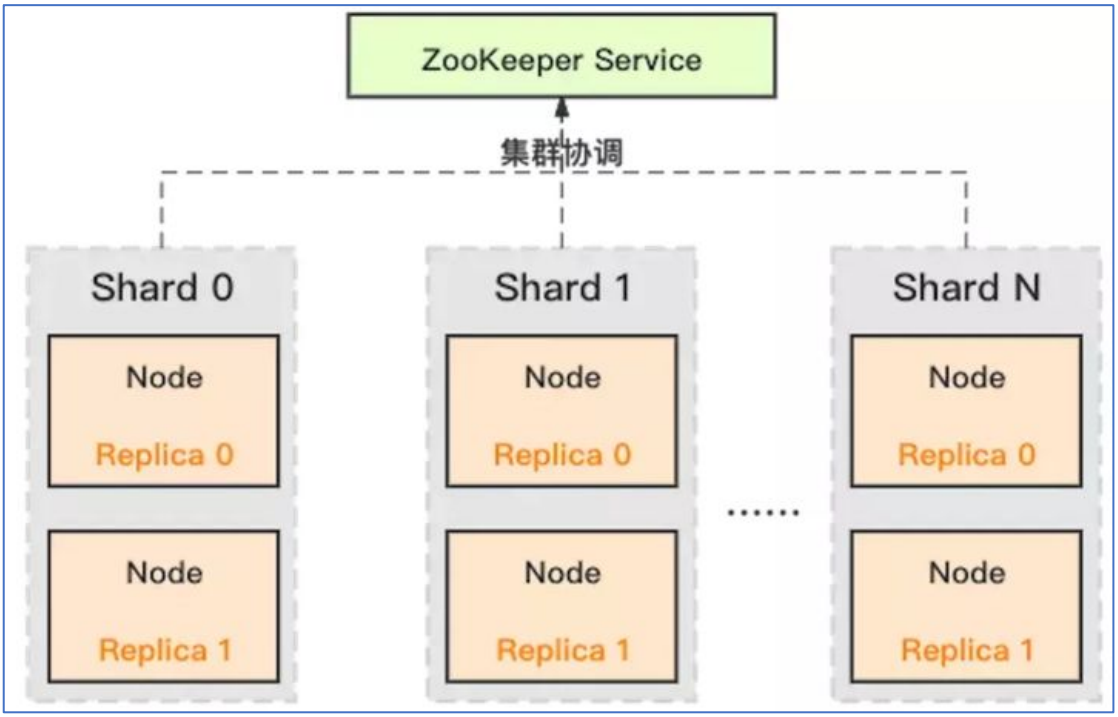
- 当分析场景中往往需要读大量行但是少数几个列时，在行存模式下，数据按行连续存储，所有列的数据都存储在一个 block 中，不参与计算的列在 IO 时也要全部读出，读取操作被严重放大。而列存模式下，只需要读取参与计算的列即可，极大的减低了 IO cost，加速了查询。
- 同一列中的数据属于同一类型，压缩效果显著。列存往往有着高达十倍甚至更高的压缩比，节省了大量的存储空间，降低了存储成本。
- 更高的压缩比意味着更小的 data size，从磁盘中读取相应数据耗时更短。
- 自由的压缩算法选择。不同列的数据具有不同的数据类型，适用的压缩算法也就不尽相同。可以针对不同列类型，选择最合适的压缩算法。
- 高压缩比 ,意味着同等大小的内存能够存放更多数据 ,系统 cache 效果更好。

4、ClickHouse 的缺点是是什么？

- 不支持事务，不支持真正的删除/更新；
- 不支持二级索引；

- join 实现与众不同；
- 不支持窗口功能；
- 元数据管理需要人为干预。

5、ClickHouse 的架构是怎样的？



ClickHouse 采用典型的分组式的分布式架构，其中：

- Shard。集群内划分为多个分片或分组（Shard 0 ... Shard N），通过 Shard 的线性扩展能力，支持海量数据的分布式存储计算。
- Node。每个 Shard 内包含一定数量的节点（Node，即进程），同一 Shard 内的节点互为副本，保障数据可靠。ClickHouse 中副本数可按需建设，且逻

辑上不同 Shard 内的副本数可不同。

- ZooKeeper Service。集群所有节点对等，节点间通过 ZooKeeper 服务进行分布式协调。

6、ClickHouse 的逻辑数据模型？

从用户使用角度看，ClickHouse 的逻辑数据模型与关系型数据库有一定的相似：一个集群包含多个数据库，一个数据库包含多张表，表用于实际存储数据。

7、ClickHouse 的核心特性？

- 列存储：列存储是指仅从存储系统中读取必要的列数据，无用列不读取，速度非常快。ClickHouse 采用列存储，这对于分析型请求非常高效。一个典型且真实的情况是，如果我们需要分析的数据有 50 列，而每次分析仅读取其中的 5 列，那么通过列存储，我们仅需读取必要的列数据，相比于普通行

存，可减少 10 倍左右的读取、解压、处理等开销，对性能会有质的影响。

- 向量化执行：在支持列存的基础上，ClickHouse 实现了一套面向 向量化处理 的计算引擎，大量的处理操作都是向量化执行的。相比于传统火山模型中的逐行处理模式，向量化执行引擎采用批量处理模式，可以大幅减少函数调用开销，降低指令、数据的 Cache Miss，提升 CPU 利用效率。并且 ClickHouse 可利用 SIMD 指令进一步加速执行效率。这部分是 ClickHouse 优于大量同类 OLAP 产品的重要因素。
- 编码压缩：由于 ClickHouse 采用列存储，相同列的数据连续存储，且底层数据在存储时是经过排序的，这样数据的局部规律性非常强，有利于获得更高的数据压缩比。此外，ClickHouse 除了支持 LZ4、ZSTD 等通用压缩算法外，还支持 Delta、DoubleDelta、Gorilla 等专用编码算法，用于进一步提高数据压缩比。
- 多索引：列存用于裁剪不必要的字段读取，而索引则用于裁剪不必要的记录读取。ClickHouse 支持丰富的索引，从而在查询时尽可能的裁剪不必要的记录读取，提高查询性能。

8、使用 ClickHouse 时有哪些注意点？

分区和索引

分区粒度根据业务特点决定，不宜过粗或过细。一般选择按天分区，也可指定为

tuple()；以单表 1 亿数据为例，分区大小控制在 10-30 个为最佳。

必须指定索引列，clickhouse 中的索引列即排序列，通过 order by 指定，一般在查询条件中经常被用来充当筛选条件的属性被纳入进来；可以是单一维度，也可以是组合维度的索引；通常需要满足高级列在前、查询频率大的在前原则；还有基数特别大的不适合做索引列，如用户表的 userid 字段；通常筛选后的数据满足在百万以内为最佳。

数据采样策略

通过采用运算可极大提升数据分析的性能。

数据量太大时应避免使用 select * 操作，查询的性能会与查询的字段大小和数量成线性变换；字段越少，消耗的 IO 资源就越少，性能就会越高。

千万以上数据集用 order by 查询时需要搭配 where 条件和 limit 语句一起使用。

如非必须不要在结果集上构建虚拟列，虚拟列非常消耗资源浪费性能，可以考虑在前端进行处理，或者在表中构造实际字段进行额外存储。

不建议在高基列上执行 distinct 去重查询，改为近似去重 uniqCombined。

多表 Join 时要满足小表在右的原则，右表关联时被加载到内存中与左表进行比较。

存储

ClickHouse不支持设置多数据目录，为了提升数据io性能，可以挂载虚拟券组，一个券组绑定多块物理磁盘提升读写性能；多数查询场景 SSD 盘会比普通机械硬盘快 2-3 倍。

9、 ClickHouse 的引擎有哪些？

ClickHouse 提供了大量的数据引擎，分为数据库引擎、表引擎，根据数据特点及使用场景选择合适的引擎至关重要。

ClickHouse 引擎分类

引擎分类	引擎名称	
数据库引擎	Ordinary/Dictionary/Memory/Lazy/MySQL	
数据表引擎	MergeTree系列	MergeTree 、ReplacingMergeTree 、SummingMergeTree 、AggregatingMergeTree CollapsingMergeTree 、VersionedCollapsingMergeTree 、GraphiteMergeTree
	Log系列	TinyLog 、StripeLog 、Log
	Integration Engines	Kafka 、MySQL、ODBC 、JDBC、HDFS
	Special Engines	Distributed 、MaterializedView、Dictionary 、Merge 、File、Null 、Set 、Join 、 URL View、Memory 、Buffer

在以下几种情况下，ClickHouse 使用自己的数据库引擎：

- 决定表存储在哪里以及以何种方式存储；
- 支持哪些查询以及如何支持；
- 并发数据访问；
- 索引的使用；
- 是否可以执行多线程请求；
- 数据复制参数。

在所有的表引擎中，最为核心的当属 MergeTree 系列表引擎，这些表引擎拥有最为强大的性能和最广泛的使用场合。对于非 MergeTree 系列的其他引擎而言，主要用于特殊用途，场景相对有限。而 MergeTree 系列表引擎是官方主推的存储引擎，支持几乎所有 ClickHouse 核心功能。

MergeTree 作为家族系列最基础的表引擎，主要有以下特点：

- 存储的数据按照主键排序：允许创建稀疏索引，从而加快数据查询速度；
- 支持分区，可以通过 PRIMARY KEY 语句指定分区字段；
- 支持数据副本；
- 支持数据采样。

10、建表引擎参数有哪些？

ENGINE：ENGINE = MergeTree()，MergeTree 引擎没有参数。

ORDER BY：order by 设定了分区内的数据按照哪些字段顺序进行有序保存。

order by 是 MergeTree 中唯一一个必填项，甚至比 primary key 还重要，因为当用户不设置主键的情况，很多处理会依照 order by 的字段进行处理。

要求：主键必须是 order by 字段的前缀字段。

如果 ORDER BY 与 PRIMARY KEY 不同，PRIMARY KEY 必须是 ORDER BY 的前缀(为了保证分区内数据和主键的有序性)。

ORDER BY 决定了每个分区中数据的排序规则;

PRIMARY KEY 决定了一级索引(primary.idx);

ORDER BY 可以指代 PRIMARY KEY, 通常只用声明 ORDER BY 即可。

PARTITION BY：分区字段，可选。如果不填：只会使用一个分区。

分区目录：MergeTree 是以列文件+索引文件+表定义文件组成的，但是如果设定了分区那么这些文件就会保存到不同的分区目录中。

PRIMARY KEY：指定主键，如果排序字段与主键不一致，可以单独指定主键字段。否则默认主键是排序字段。可选。

SAMPLE BY：采样字段，如果指定了该字段，那么主键中也必须包含该字段。比如 SAMPLE BY intHash32(UserID) ORDER BY (CounterID, EventDate, intHash32(UserID))。可选。

TTL：数据的存活时间。在 MergeTree 中，可以为某个列字段或整张表设置 TTL。当时间到达时，如果是列字段级别的 TTL，则会删除这一列的数据；如果是表级别的 TTL，则会删除整张表的数据。可选。

SETTINGS：额外的参数配置。可选。